密 级：□公开资料 □内部资料 □保密资料 □机密资料

视频图像质量检测系统说明文档

文件编号			
文件状态	[]草稿 [] 正式发布 []正在修改		
当前版本	Vx.x.x		
拟 制		日期	xxxx-xx-xx
审 核		日期	
批 准		日期	
参与编制			
发布日期	年 月 日		
参考模板及版本号			

文档控制

1） 文档更新记录

A – 增加　M – 修订　D – 删除

日期	更新人	版本	变更类型（A*M*D）	备注
2014 年 6 月 25 日	黄梦延	初版	A	
2015 年 1 月 8 日	黄梦延	增加信号丢失/画面冻结	A+M	
2016 年 3 月 7 日	周明辉	修改图像模糊和图像噪声检测	M	

2） 文档审核记录

日期	审核人	职务	备注

3） 文档发行范围

分发单位	说明
四川华雁信息产业技术股份有限公司	

前言

概述

本节介绍了本文档的内容，对应的产品系统，适用的读者范围。

产品系统

与本文档对应的系统是华雁视频图像质量检测系统。

读者对象

本文档主要适用以下工程师：

- 该检测系统的测试工程师
- 该检测系统的维护工程师

目 录

1.概述

1.1 概述

华雁视频图像质量检测系统是四川华雁信息产业股份有限公司提供的检测视频图像质量的相应模块。

1.2 参考域说明

1.2.1 API 参考域

本说明文档适用了 8 个参考描述 API 的相关信息，它们的作用如下表 1-1 所示。

表 1-1 API 参考域说明

参考域	含义
1 目的	描述相应 API 的主要功能
2 语法	列出调用 API 需要包括的头文件以及 API 的原形声明
3 参数	列出了 API 中调用的参数，参数类型以及参数属性说明
4 描述	简要的描述 API 的工作过程
5 返回值	描述了 API 返回的值及其含义
6 需求	描述 API 包含的头文件以及其需要连接的库
7 注意	使用 API 时需要注意的点
8 举例	描述如何使用 API

1.2.2 数据类型参考域

本说明文档采用了 5 个参考域描述数据类型的相关信息，它们的作用如下表 1-2 所示。

表 1-2 参考域描述数据类型

参考域	说明含义
1 说明	描述该数据类型的主要功能
2 含义	列出数据类型的定义
3 成员	列出该数据类型包含的成员
4 注意事项	列出了使用该数据类型的注意点
5 相关数据类型以及接口	列出了与该数据类型相关的数据类型以及接口

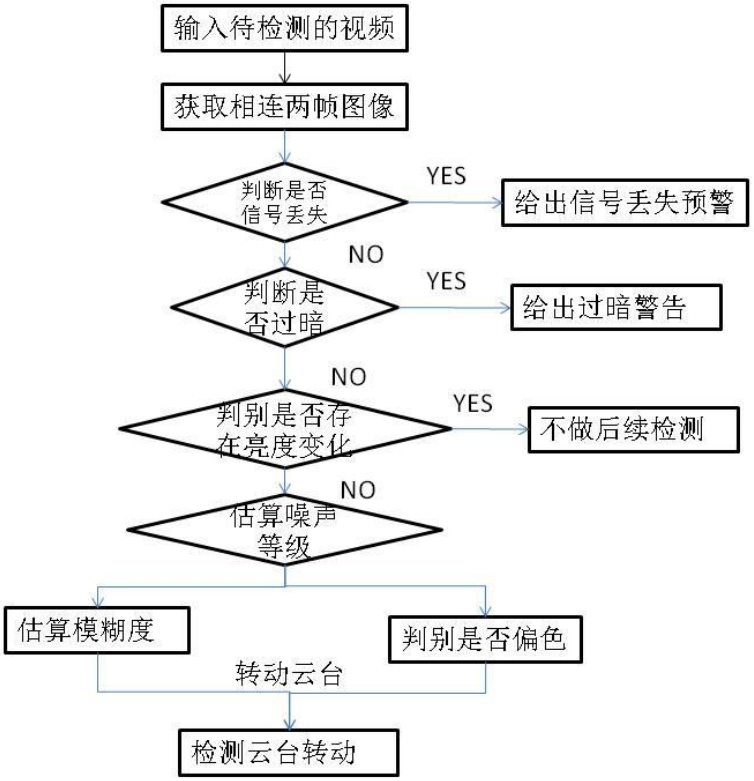
2. 华雁视频图像质量检测系统简介

华雁视频图像质量检测系统主要是在实时监控时检测监控视频的图像质量，并对信号丢失，视频图像模糊，视频图像偏色，视频过噪，视频图像亮度异常，视频遮挡，PTZ 转动异常等异常情况给出预警。

2.1 视频图像质量检测系统流程

华雁视频图像质量检测系统的主要流程如下图 2-1 所示。

图 2-1 视频图像质量检测系统流程



2.2 视频图像质量检测系统包含部分

华雁视频图像质量检测系统主要包括七个部分，分别是视频的信号丢失检测，视频图像的噪声等级检测，视频图像的模糊度检测，视频图像的亮度异常检测，视频图像的偏色检测，视频图像的画面冻结异常检测以及云台转动异常检测。目前提供的库包括了视频图像的模糊度等级检测，视频图像的噪声等级检测，视频图像的亮度异常检测，视频的偏色检测四个部分。视频的信号丢失异常检测，视频画面冻结异常以及云台转动异常的库待添加。

3.API 参考

3.1 HYIQ_MemMgrCreate

参考域	含义	
1 目的	调用此接口函数，预先申请一定的内存	
2 语法	调用需要包含“HY_IMAGEQUALITY.H”文件 调用语法如下 HYIQ_MemMgrCreate(MVoid * pMem, MLong lMemSize);	
3 参数	MVoid * pMem	需要分配的内存指针
	MLong lMemSize	需要分配的内存大小
4 描述	调用该函数，给句柄分配一定大小的内存	
5 返回值	0	接口运行正常
	-1	接口运行异常
6 需求	调用需要包含“HY_IMAGEQUALITY.H”文件 需要包含 ImageQuality.lib 的库文件	
7 注意	需预先估算可能要用到的内存大小，不然有可能会导致后续程序内存不足。	
8 举例	#define WORK_BUFFER 2560*1920 MHandle hMemMgr = HYIQ_MemMgrCreate(pMem,WORK_BUFFER);	

3.2 HYIQ_Init

参考域	含义	
1 目的	调用此接口初始化	
2 语法	调用需要包含“HY_IMAGEQUALITY.H”文件 调用语法如下 HYIQ_Init(MHandle hMemMgr,MHandle *pIQreg);	
3 参数	MHandle hMemMgr	输入的操作句柄，如果值为 MNULL，则在系统内部进行内存的分布。 如果输入的值不是 MNULL，则其调用的内存是 HYIQ_MemMgrCreate 生成的。
	MHandle *pIQreg	输出的内部操作句柄
4 描述	根据外部输入的操作句柄分配内部操作句柄的内存	
5 返回值	0	接口运行正常
	-1	接口运行异常
6 需求	调用需要包含“HY_IMAGEQUALITY.H”文件 需要包含 ImageQuality.lib 的库文件	
7 注意	无	
8 举例	MHandle hMemMgr = MNull; MHandle HYIQHandle=MNull; HYIQ_Init(hMemMgr,&HYIQHandle);	

3.3 HYIQ_LOST

参考域	含义

1 目的	调用此接口判别函数是否存在信号丢失或者画面冻结	
2 语法	调用需要包含“HY_IMAGEQUALITY.H”文件 调用语法如下： HYIQ_LOST(MHandle hHandle,IQ_PIMAGES pImagesrc1,IQ_PIMAGES pImagesrc2,HYIQ_PTOutParam ptOutParam);	
3 参数	MHandle hMemMgr	输入的操作句柄
	IQ_PIMAGES pImagesrc1	输入的原始图像 1
	IQ_PIMAGES pImagesrc2	输入的原始图像 2
	HYIQ_PTOutParam ptOutParam	输出的参数
4 描述	使用此接口可以得到图像的噪声等级估计	
5 返回值	0	接口运行正常
	-1	接口运行错误
6 需求	调用需要包含“HY_IMAGEQUALITY.H”文件 需要包含 ImageQuality.lib 的库文件	
7 注意	输入的图像需为 IQ_PIMAGES，具体格式见第 4 章，输入的两幅图像需要为前后帧图像	

8 举例	下文中的img1，img2均为借用外部opencv库获取的图像。
	IQ_IMAGES imgiq1={0};
	IQ_IMAGES imgiq2={0};
	HYIQ_TOutParam p1={0};
	imgiq1.lHeight=img1->height;
	imgiq1.lWidth=img1->width;
	imgiq1.lPixelArrayFormat=HY_IMAGE_BGR;
	imgiq1.pixelArray.chunky.lLineBytes=img1->widthStep;
	imgiq1.pixelArray.chunky.pPixel=(unsigned char*)(img1->imageData);
	imgiq2.lHeight=img2->height;
	imgiq2.lWidth=img2->width;
	imgiq2.lPixelArrayFormat=HY_IMAGE_BGR;
	imgiq2.pixelArray.chunky.lLineBytes=img2->widthStep;
	imgiq2.pixelArray.chunky.pPixel=(unsigned char *)(img2->imageData);
	HYIQ_LOST(HYIQHandle,&imgiq1,&imgiq2,&p1);	

3.4 HYIQ_NOISE

参考域	含义	
1 目的	调用此接口获得图像的噪声等级	
2 语法	调用需要包含“HY_IMAGEQUALITY.H”文件 调用语法如下： HYIQ_NOISE(MHandle hMemMgr,IQ_PIMAGES pImage,IQ_PIMAGES pImage1,HYIQ_PTOutParam ptOutParam);	
3 参数	MHandle hMemMgr	输入的操作句柄
	IQ_PIMAGES pImage	输入的原始图像 1
	IQ_PIMAGES pImage1	输入的原始图像 2
	HYIQ_PTOutParam ptOutParam	输出的参数
4 描述	使用此接口可以得到图像的噪声等级估计	

5 返回值	0	接口运行正常
	-1	接口运行错误
6 需求	调用需要包含“HY_IMAGEQUALITY.H”文件 需要包含 ImageQuality.lib 的库文件	
7 注意	输入的图像需为 IQ_PIMAGES，具体格式见第 4 章，输入的两幅图像需要为前后帧图像	

8 举例	下文中的img1，img2均为借用外部opencv库获取的图像。
	IQ_IMAGES imgiq1={0};
	IQ_IMAGES imgiq2={0};
	HYIQ_TOutParam p1={0};
	imgiq1.lHeight=img1->height;
	imgiq1.lWidth=img1->width;
	imgiq1.lPixelArrayFormat=HY_IMAGE_BGR;
	imgiq1.pixelArray.chunky.lLineBytes=img1->widthStep;
	imgiq1.pixelArray.chunky.pPixel=(unsigned char*)(img1->imageData);
	imgiq2.lHeight=img2->height;
	imgiq2.lWidth=img2->width;
	imgiq2.lPixelArrayFormat=HY_IMAGE_BGR;
	imgiq2.pixelArray.chunky.lLineBytes=img2->widthStep;
	imgiq2.pixelArray.chunky.pPixel=(unsigned char *)(img2->imageData);
	HYIQ_NOISE(HYIQHandle,&imgiq1,&imgiq2,&p1);	

3.5 HYIQ_BLUR

参考域	含义	
1 目的	调用此接口对图像进行清晰度检测	
2 语法	调用需要包含“HY_IMAGEQUALITY.H”文件 调用语法如下： HYIQ_CLEAR(MHandle hMemMgr,IQ_PIMAGES pImage,IQ_PIMAGES pImage1,HYIQ_PTOutParam ptOutParam);	
3 参数	MHandle hMemMgr	输入的操作句柄
	IQ_PIMAGES pImage	输入的原始图像 1
	IQ_PIMAGES pImage1	输入的原始图像 2
	HYIQ_PTOutParam ptOutParam	输出的参数
4 描述	使用此接口得到图像的清晰度检测（清晰，不清晰，严重不清晰）	
5 返回值	0	接口运行正常
	-1	接口运行错误
6 需求	调用需要包含“HY_IMAGEQUALITY.H”文件 需要包含 ImageQuality.lib 的库文件	
7 注意	输入的图像需为 IQ_PIMAGES，具体格式见第 4 章，输入的两幅图像需要为前后帧图像	
8 举例	下文中的img1，img2均为借用外部opencv库获取的图像。 IQ_IMAGES imgiq1={0}; IQ_IMAGES imgiq2={0};	

	HYIQ_TOutParam p1={0};
	imgiq1.lHeight=img1->height;
	imgiq1.lWidth=img1->width;
	imgiq1.lPixelArrayFormat=HY_IMAGE_BGR;
	imgiq1.pixelArray.chunky.lLineBytes=img1->widthStep;
	imgiq1.pixelArray.chunky.pPixel=(unsigned char*)(img1->imageData);
	imgiq2.lHeight=img2->height;
	imgiq2.lWidth=img2->width;
	imgiq2.lPixelArrayFormat=HY_IMAGE_BGR;
	imgiq2.pixelArray.chunky.lLineBytes=img2->widthStep;
	imgiq2.pixelArray.chunky.pPixel=(unsigned char *)(img2->imageData);
	HYIQ_BLUR(HYIQHandle,&imgiq1,&imgiq2,&p1);

3.6 HYIQ_BRIGHT

参考域	含义	
1 目的	调用此接口检测视频图像的亮度异常	
2 语法	调用需要包含“HY_IMAGEQUALITY.H”文件 调用语法如下： HYIQ_BRIGHT(MHandle hMemMgr,IQ_PIMAGES pImage, IQ_PIMAGES pImage,1 HYIQ_PTOutParam ptOutParam);	
3 参数	MHandle hMemMgr	操作句柄
	IQ_PIMAGES pImage	输入图像
	IQ_PIMAGES pImage1	输入图像
	HYIQ_PTOutParam ptOutParam	输出参数
4 描述	检测视频图像是否亮度异常	
5 返回值	0	接口运行正常
	-1	接口运行错误
6 需求	调用需要包含“HY_IMAGEQUALITY.H”文件 需要包含 ImageQuality.lib 的库文件	
7 注意	输入的图像需为 IQ_PIMAGES，具体格式见第 4 章	

8 举例	下文中的img1,img2为借用外部opencv库获取的图像。
	IQ_IMAGES imgiq1={0};
	IQ_IMAGES imgiq2={0};
	HYIQ_TOutParam p1={0};
	imgiq1.lHeight=img1->height;
	imgiq1.lWidth=img1->width;
	imgiq1.lPixelArrayFormat=HY_IMAGE_BGR;
	imgiq1.pixelArray.chunky.lLineBytes=img1->widthStep;
	imgiq1.pixelArray.chunky.pPixel=(unsigned char*)(img1->imageData);
	imgiq1.lHeight=img1->height;
	imgiq1.lWidth=img1->width;
	imgiq1.lPixelArrayFormat=HY_IMAGE_BGR;
	imgiq1.pixelArray.chunky.lLineBytes=img1->widthStep;
	imgiq1.pixelArray.chunky.pPixel=(unsigned char*)(img1->imageData);	

	HYIQ_BRIGHT(HYIQHandle,&imgiq1,&imgiq2,&p1);

3.7 HYIQ_CAST

参考域	含义	
1 目的	调用此接口检测图像的异常	
2 语法	调用需要包含“HY_IMAGEQUALITY.H”文件 调用语法如下： HYIQ_CAST(MHandle hMemMgr,IQ_PIMAGES pImage, HYIQ_PTOutParam ptOutParam);	
3 参数	MHandle hMemMgr	操作句柄
	IQ_PIMAGES pImage	输入图像
	HYIQ_PTOutParam ptOutParam	输出参数
4 描述	检测视频图像是否存在偏色	
5 返回值	0	接口运行正常
	-1	接口运行错误
6 需求	调用需要包含“HY_IMAGEQUALITY.H”文件 需要包含 ImageQuality.lib 的库文件	
7 注意	输入的图像需为 IQ_PIMAGES，具体格式见第 4 章	

8 举例	下文中的img1为借用外部opencv库获取的图像。
	IQ_IMAGES imgiq1={0};
	HYIQ_TOutParam p1={0};
	imgiq1.lHeight=img1->height;
	imgiq1.lWidth=img1->width;
	imgiq1.lPixelArrayFormat=HY_IMAGE_BGR;
	imgiq1.pixelArray.chunky.lLineBytes=img1->widthStep;
	imgiq1.pixelArray.chunky.pPixel=(unsigned char*)(img1->imageData);
	HYIQ_CAST(HYIQHandle,&imgiq2,&p1);	

3.8 HYIQ_PTZ

该接口主要检测云台是否运动正常，此部分函数需要添加。

3.9 HYIQ_RESULT

参考域	含义	
1 目的	调用此接口主要是将输出结果和用户设置的参数进行比较，输出图像质量各个方面的状态	
2 语法	调用需要包含“HY_IMAGEQUALITY.H”文件 调用语法如下： HYIQ_RESULT(MHandle hHandle,HYIQ_PTOutParam ptOutParam,PCustomParam custom,pHYIQ_result result);	
3 参数	Handle	操作句柄
	HYIQ_PTOutParam ptOutParam	输入的结果
	PCustomParam custom	用户设置的结果，可以全为 0

	pHYIQ_result result	输出参数，包含视频图像质量各个方面的状态
4 描述	输出图像质量各个方面的状态	
5 返回值	0	接口运行正常
	-1	接口运行错误
6 需求	调用需要包含“HY_IMAGEQUALITY.H”文件 需要包含 ImageQuality.lib 的库文件	
7 注意	输入的参数格式具体见第四章	
8 举例	其中output由之前的函数获得，custopar由用户设置，可以全为0，testresult为最后输出的结果。 HYIQ_RESULT(HYIQHandle,&output,&custopar,&testresult);	

3.10 HYIQ_Uninit

参考域	含义	
1 目的	调用此接口释放内存	
2 语法	调用需要包含“HY_IMAGEQUALITY.H”文件 调用语法如下 HYIQ_Uninit(MHandle HYIQhandle);	
3 参数	MHandle HYIQhandle	待释放的内存
4 描述	释放分配的内存	
5 返回值	0	接口运行正常
	-1	接口运行异常
6 需求	调用需要包含“HY_IMAGEQUALITY.H”文件 需要包含 HY_IMAGEQUALITY.Lib 的库文件	
7 注意	无	
8 举例	MHandle HYIQHandle=MNull; HYIQ_Uninit(HYIQHandle);	

3.11 HYIQ_MemMgrDestroy

参考域	含义	
1 目的	调用此接口释放由 HYIQ_MemMgrCreate 申请的内存	
2 语法	调用需要包含“HY_IMAGEQUALITY.H”文件 调用语法如下 HYIQ_MemMgrDestroy(MHandle hMemMgr);	
3 参数	MHandle hMemMgr	待释放的内存
4 描述	释放分配的内存	
5 返回值	0	接口运行正常
	-1	接口运行异常
6 需求	调用需要包含“HY_IMAGEQUALITY.H”文件 需要包含 HY_IMAGEQUALITY.Lib 的库文件	
7 注意	需要在 HYIQ_MemMgrCreate 之后使用	
8 举例	HYIQ_MemMgrDestroy(hMemMgr);	

4. 参数说明

4.1 输入参数说明

4.1.1 输入图像格式说明

参数名称：IQ_IMAGES, *IQ_PIMAGES;

参数定义：

```
typedef struct {

    MLong       lWidth;             // Off-screen width
    MLong       lHeight;            // Off-screen height
    MLong       lPixelArrayFormat;  // Format of pixel array
    union
    {
        struct
        {
            MLong lLineBytes;
            MVoid *pPixel;
        } chunky;
        struct
        {
            MLong lLinebytesArray[4];
            MVoid *pPixelArray[4];
        } planar;
    } pixelArray;
} IQ_IMAGES, *IQ_PIMAGES;
```

参数成员：

lWidth： 图像的宽度

lHeight：图像的高度

lPixelArrayFormat：图像的格式，例如BGR,GRAY,YUYV等

pixelArray：为图像的存储形式。

Chunky： 图像按chunky格式存储，lLineBytes指图像的行长，*pPixel为像素指针

Planar：图像按planar格式存储，lLinebytesArray[4]为图像的行长，*pPixelArray[4]为图像的像素指针。

注意事项：需要将读取的图像转换为该图像格式，为接口函数调用。

相关接口：

HYIQ_NOISE,HYIQ_BLUR,HYIQ_CAS,HYIQ_BRIGHT,HYIQ_FROZEN,HYIQ_FROZEN,HYIQ_LOST

4.2 输出参数格式说明

4.2.1 参数名称：HYIQ_TOutParam,*HYIQ_PTOutParam;

参数定义

```
typedef struct
{
        struct
    {
        MFloat CLEARLEVEL; /// \blurlevel
        MLong isDetct;//0没有检测  1检测过
    }CLEAR;
  struct
  {
      MFloat sigma1;
      MFloat sigma2;
  }Sigma;
        struct
    {
        MFloat NOISELEVEL; /// \noiselevel
        MLong isDetct;//0没有检测  1检测过
    }NOISE;
```

```
    struct
    {
      MFloat BrightLevel1;
      MFloat BrightLevel2;
      MFloat Brightderiv;/// \luminace derivation
      MLong grayNum;//计算在Bright函数中
      MLong isDetct;//0没有检测  1检测过
    }Bright;
    struct
    {
          MFloat Rcastratio;
          MFloat Gcastratio;
          MFloat Bcastratio;
          MLong flag;
          MLong isDetct;//0没有检测  1检测过
    }ImgCast;
       struct
       {
            MLong flag;    /// \while flag=0, signal is normal,flag=1,signal lost
            MLong isDetct;//0没有检测  1检测过
       }SignalLost;
       struct
       {
            MLong flag;    /// \while flag=0, signal is normal,flag=1,image frozen
            MLong isDetct;//0没有检测  1检测过
       }ImgFrozen;
}HYIQ_TOutParam,*HYIQ_PTOutParam;
```

参数说明：

CLEAR：图像清晰度等级

Sigma:图像的 3 次再模糊之后估算出来的 sigma 值，由 3 个值决定最后的 blurlevel。

NOISELEVEL：图像的噪声等级

BRIGHT：图像的亮度偏差，偏差的绝对值越大，图像的亮度越异常。可以根据实际环境设置阈值。

BrightLevel1:前一帧图像的亮度等级

BrightLevel2：后一阵图像的亮的等级

Brightderiv：两帧图像的亮度偏差

ImgCast：图像的偏色相关信息，Rcastratio 表示 R 通道偏色程度，Gcastratio 表示 G 通道偏色程度，Bcastratio 表示 B 通道偏色相关程度，flag=1 时图像偏色，flag=0 时图像正常。

SignalLost：信号是否丢失；flag=0, 信号没有丢失,flag=1, 信号丢失

ImgFrozen：图像是否冻结； flag=0, 图像正常,flag=1,图像冻结

isDetct：算法是否调用检测

注意事项：

无

接口函数：

HYIQ_NOISE,HYIQ_CLEAR,HYIQ_CAST,HYIQ_BRIGHT,HYIQ_FROZEN,HYIQ_LOST

4.2.2 参数名称:HYIQ_result,*pHYIQ_result;

参数定义: typedef struct

```
{
    MLong Noisestatus;
```

```
    MLong lightstatus;
    MLong clearstatus;
    MLong caststatus;
     MLong signalstatus;
    MLong imgFrozenstatus;  ///图像冻结判别状况
}HYIQ_result,*pHYIQ_result;
```

参数说明

Noisestatus：图像的噪声等级，分为3个级别，分别为无噪声，噪声，重度噪声。

```
#define IMAGE_UNNOISE     904    无噪声
#define IMAGE_MEDNOISE    905    中度噪声
#define IMAGE_HEAVENOISE  906    重度噪声
```

Lightstatus：图像的亮度异常情况，分别为过暗和过亮。

```
#define IMAGE_TOODARK     907    图像过亮
#define IMAGE_TOOLIGHT    908    图像过暗
```

blurstatus：图像的模糊等级，分为4个级别，分别为不模糊，轻度模糊，中度模糊和重度模糊。

```
#define IMAGE_CLEAR      900    图像不模糊
#define IMAGE_NOTCLEAR    901    图像轻度模糊
#define IMAGE_HEVNOTCLEAR    902    图像中度模糊
```

Caststatus：图像的偏色情况

```
#define IMAGE_CAST        909    图像偏色
```

Signalstatus:视频信号问题，主要问视频信号丢失以及视频画面冻结

```
#define IMAGE_SIGNALLOST  911
#define IMAGE_FROZEN      912
```

注意事项：

无

接口函数：HYIQ_RESULT

4.2.3 参数名称：CustomParam,*PCustomParam;

参数定义：

```
typedef struct
{
    struct
    {
        MFloat ClearLevellowthresh;/// if blur level is less than this
threshhold,the image is not blur
        MFloat ClearLevelhighthresh;///if blur level is larger than
ClearLevellowthresh,but less than this ,the image is a  blurred;
    }ClearLevel;
    struct
    {
        MLong NoiseLevellowthresh;/// if blur level is less than this
threshhold,the image is not too noise
        MLong Noiselevelhighthresh;///if blur level is larger than
BlurLevellowthresh,but less than this ,the image is a litter noise;
    }Noiselevel;
    struct
    {
        MFloat darkthresh; ///when small than this thresh,the image is too dark
        MFloat lightthresh;/// when larger than this thresh the image is too light

    }Light;
    MFloat castratio;///when larger than this ratio,the image is cast;

    }CustomParam,*PCustomParam;
```

参数说明:

BlurLevellowthresh：当小于这个低阈值的时候，图像清晰

Blurlevelhighthresh;当大于 BlurLevellowthresh 且小于 Blurlevelhighthresh 时，图像模糊。当大于 Blurlevelhighthresh 时，图像重度模糊。

NoiseLevellowthresh：当小于 NoiseLevellowthresh 时，图像无噪声。

Noiselevelhighthresh:当大于 NoiseLevellowthresh 且小于 Noiselevelhighthresh，图像存在噪声；当大于 NoiseLevellowthresh 时，图像重度噪声。

Darkthresh：当亮度偏差小于这个阈值的时候，图像过暗。

Lightthresh：当亮度偏差大于这个阈值的时候，图像过亮。

Castratio：当颜色偏差大于这个阈值的时候，图像偏色。

注意事项：

若这个参数输入为 0 的时候，在 HYIQ_RESULT 函数下调用默认参数。

BlurLevellowthresh：1.4

Blurlevelhighthresh;20

NoiseLevellowthresh：3

Noiselevelhighthresh：20

Darkthresh：-0.45

Lightthresh：0.25

Castratio：0.35

接口函数：HYIQ_RESULT

4.3 返回错误值类型说明

错误名称	说明	返回值
HYIQ_ERR_NONE	无错误	0
HYIQ_ERR_UNKNOWN	未知错误	-1
HYIQ_ERR_IMAGE_FORMAT	输入图像类型错误	-101
HYIQ_ERR_IMAGE_SIZE_UNMATCH	输入图像的尺寸不相符合	-102
HYIQ_ERR_ALLOC_MEM_FAIL	内存分配错误	-201
HYIQ_ERR_NO_FIND	未找到输入	-901

5. 华雁视频图像质量检测要求

根据第三章的内容设置好各个函数的输入输出接口，初始化各个函数。对实时视频进行检测，输出相应的参数，根据实时监控场景的相关环境设置合理阈值，估算出最后的检测正确度。

5.1 图像的亮度异常检测

由于亮度异常检测将直接影响后续的检测，例如，如果图像过暗的时候，将导致图像偏色检测异常或者模糊度估计的时候误差较大，因此，需要优先检测亮度异常情况。由于噪声估计部分采取的检测方法是帧差，因此，需要保证两帧图像之间无明显亮度变化，之后才进行图像的噪声等级估计。

5.2 图像的噪声等级估计

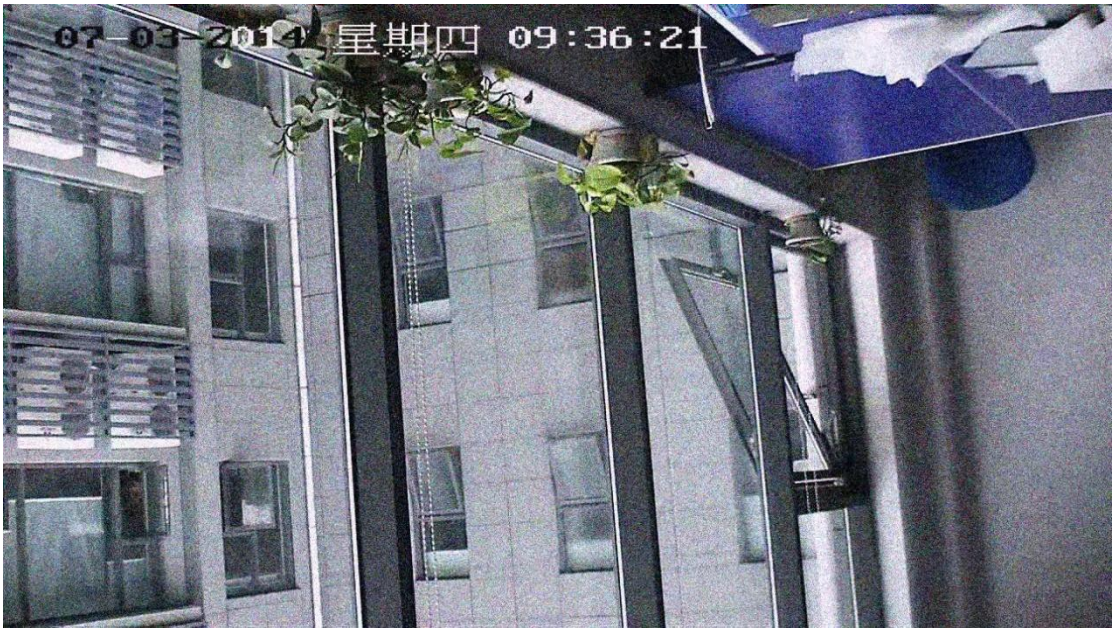
由于图像的噪声等级估计采取的是帧差的方法，所以检测前需要保证两帧图像之间无明显的亮度变化，并且进行噪声等级估计的时候，希望云台能够固定不动。

检测的时候我们将图像的噪声等级分为 3 类，分别是低噪声等级，中度噪声等级，重度噪声等级。

低度噪声等级是指，存在噪声，但是噪声的存在不影响实际图像内容。估算出来的噪声等级范围为 2-5。

低度噪声等级图像

中度噪声等级是指，存在噪声，并且较为明显，估算出来的噪声等级范围为 5-20。



中度噪声等级图像

重度噪声等级是指，存在噪声，并且较为明显，会对后续检测有所影响，估算出来的噪声等级范围为大于 20

重度噪声等级图像

5.3 图像的清晰估计

由于图像过暗的时候，将导致图像内容不清晰，因此，在进行清晰检测的之前，需要先进行图像的亮度异常检测。如果图像过暗，则不进行图像清晰检测。由于球机在转动过程中会存在失焦再对焦的过程，因此进行图像清晰检测的时候需要云台固定。并且，如果球机拍

摄到的画面中存在运动物体，则球机可能由于对焦而产生短暂的模糊，因此，在检测的时候，可以设置一定的时间间隔，在时间间隔内，如果图像的模糊度等级持续大于阈值，则预警。

本测试中将图像清晰分为三个等级，分别为图像清晰，图像不清晰和图像重度不清晰。

图像清晰：图像中存在较轻微的不清晰或轻微不清晰，不影响实际图像内容，估算出来的值小于 5。

图像轻度模糊

图像不清晰：图像中存在较为明显的不清晰，估算出来的值为 5-20。

图像中度模糊

图像重度不清晰：图像中存在明显重度不清晰，估算值大于 20。

图像重度模糊

5.4 图像的偏色检测

由于视频图像过亮的时候会导致图像的颜色信息丢失，因此在进行视频偏色检验之前，需要检测视频的亮度信息，当视频过暗活着过亮的时候，不进行偏色检验。

5.5 视频信号丢失/画面冻结检测

视频信号丢失是指由于信号传输过程出现问题，导致视频显示端无信号，出现蓝屏或者单一屏幕，或者出现”视频无信号”等相关 OSD 的时候的情况。如下图所示：

信号丢失，屏幕单一颜色

信号丢失，简单指定 OSD

画面冻结是指信号传输问题，使显示的图像冻结，无法实时更新检测画面。

6. 检测 Demo 代码

以下检测代码只能读取jpeg格式的文件。

```
// testBed.cpp : main project file.

#include"amcomdef.h"
#include"HY_IMAGEQUALITY.h"

#include <stdio.h>
#include <windows.h>
#include <stdlib.h>
#include <string.h>

#include <ctype.h>

#include <opencv/cv.h>
#include <opencv/cxcore.h>
#include <opencv/highgui.h>

#define WORK_BUFFER 2560*1920*5
#define STATIC_MEM
#define JLINE_BYTES(Width, BitCt)       (((long)(Width) * (BitCt) + 31) / 32 * 4)

int main(int argc, char* argv[])
{
    MHandle HYIQHandle=MNull;
#ifdef STATIC_MEM
    MVoid *pMem=malloc(WORK_BUFFER);
    MHandle hMemMgr = HYIQ_MemMgrCreate(pMem,WORK_BUFFER);
#else
    MHandle hMemMgr = MNull;
#endif

    IQ_IMAGES imgiq1={0};
    IQ_IMAGES imgiq2={0};
    IQ_IMAGES imgiq1_={0};
    IQ_IMAGES imgiq2_={0};

    HYIQ_TOutParam p1={0};
    CustomParam custopar={0};
    HYIQ_result testresult={0};
    //IMQU *pImqu=NULL;

    FILE *fp;
```

```
IplImage *pImage1;//视频 前一帧
IplImage *pImage2;//视频 当前帧

WIN32_FIND_DATA FindFileData;
HANDLE hFind;
char filePath[MAX_PATH]="E:\\testdata\\";
char filePath_const[MAX_PATH];
char fileName1[MAX_PATH];
char fileName2[MAX_PATH];

int nImgnum;
//统计检测结果
int nNotClearNum;
int nHeavyNotClearNum;
int nNoiseNum;
int nHeavyNoiseNum;
int nTooLightNum;
int nTooDrakNum;
int nCastNum;
int nFrozemNum;
int nLostNum;
int nNormNum;
int nLabelNorm;

int k;
int bufSize;

nImgnum=0;
nNotClearNum=0;
nHeavyNotClearNum=0;
nNoiseNum=0;
nHeavyNoiseNum=0;
nTooLightNum=0;
nTooDrakNum=0;
nCastNum=0;
nFrozemNum=0;
nLostNum=0;
nNormNum=0;
fp=fopen("..//result.txt","w");
HYIQ_Init(hMemMgr,&HYIQHandle,2);
//cvNamedWindow("img2",1);

strcpy(filePath_const,filePath);
strcat(filePath,"\*.jpeg");
```

```
    hFind = FindFirstFile(filePath, &FindFileData);
    if (hFind == INVALID_HANDLE_VALUE)
    {
        printf ("FindFirstFile failed (%d)\n", GetLastError());
        return -2;
    }

    memset(fileName1,0,MAX_PATH);
    strcpy(fileName1,filePath_const);
    strcat(fileName1,FindFileData.cFileName);
    fprintf(fp,"%s\n",fileName1);
    while (FindNextFile(hFind, &FindFileData))
    {
        nImgnum++;
        nLabelNorm=0;

        if (nImgnum==1)
        {
            memset(fileName2,0,MAX_PATH);
            strcpy(fileName2,filePath_const);
            strcat(fileName2,FindFileData.cFileName);
            fprintf(fp,"%s\n",fileName2);
        }
        else
        {
            memset(fileName1,0,MAX_PATH);
            strcpy(fileName1,filePath_const);
            strcat(fileName1,FindFileData.cFileName);
            fprintf(fp,"%s\n",fileName1);
            FindNextFile(hFind, &FindFileData);
            memset(fileName2,0,MAX_PATH);
            strcpy(fileName2,filePath_const);
            strcat(fileName2,FindFileData.cFileName);
            fprintf(fp,"%s\n",fileName2);
        }

        //从文件中读取图像
        pImage1 = cvLoadImage(fileName1/*"D:\\1\\8_0.jpeg"*/,
CV_LOAD_IMAGE_UNCHANGED);
        pImage2 = cvLoadImage(fileName2/*"D:\\1\\8_1.jpeg"*/,
CV_LOAD_IMAGE_UNCHANGED);

        imgiq1.lHeight=pImage1->height;
        imgiq1.lWidth=pImage1->width;
```

```
        imgiq1.lPixelArrayFormat=HY_IMAGE_BGR;

    imgiq1.pixelArray.chunky.lLineBytes=JLINE_BYTES(pImage1->width,pImage1->depth*pI
mage1->nChannels);
        imgiq1.pixelArray.chunky.pPixel=pImage1->imageData;

        imgiq2.lHeight=pImage2->height;
        imgiq2.lWidth=pImage2->width;
        imgiq2.lPixelArrayFormat=HY_IMAGE_BGR;

    imgiq2.pixelArray.chunky.lLineBytes=JLINE_BYTES(pImage2->width,pImage2->depth*pI
mage2->nChannels);
        imgiq2.pixelArray.chunky.pPixel=pImage2->imageData;

        p1.CLEAR.CLEARLEVEL=0;
        p1.CLEAR.isDetct=0;
        p1.Bright.BrightLevel1=0;
        p1.Bright.BrightLevel2=0;
        p1.Bright.Brightderiv=0;
        p1.Bright.isDetct=0;
        p1.ImgCast.flag=0;
        p1.ImgCast.isDetct=0;
        p1.NOISE.NOISELEVEL=0;
        p1.NOISE.isDetct=0;
        p1.SignalLost.flag=0;
        p1.SignalLost.isDetct=0;
        p1.ImgFrozen.flag=0;
        p1.ImgFrozen.isDetct=0;

        testresult.clearstatus=0;
        testresult.caststatus=0;
        testresult.lightstatus=0;
        testresult.Noisestatus=0;
        testresult.signalLoststatus=0;
        testresult.imgFrozenstatus=0;

        HYIQ_CUTIMAGE(&imgiq1_,&imgiq1);
        HYIQ_CUTIMAGE(&imgiq2_,&imgiq2);

        //pImqu=(IMQU *)HYIQHandle;
        HYIQ_LOST(HYIQHandle,&imgiq1,&imgiq2,&p1);
        p1.SignalLost.isDetct=1;
        HYIQ_BRIGHT(HYIQHandle,&imgiq1_,&imgiq2_,&p1);//计算亮度均值和均值差
        p1.Bright.isDetct=1;
```

```
if (p1.Bright.Brightderiv<2&&p1.SignalLost.flag==0)
{
	HYIQ_NOISE(HYIQHandle,&imgiq1_,&imgiq2_,&p1);
	p1.NOISE.isDetct=1;
	if (p1.Bright.BrightLevel1>-0.45&&p1.Bright.BrightLevel2>-0.45)//过暗
	{
		HYIQ_CLEAR(HYIQHandle,&imgiq1_,&imgiq2_,&p1);//过暗不检测
		p1.CLEAR.isDetct=1;
		HYIQ_CAST(HYIQHandle,&imgiq1_,&p1);//偏色　过亮或过暗不检测
		p1.ImgCast.isDetct=1;
	}
}

HYIQ_RESULT(HYIQHandle,&p1,&custopar,&testresult);
switch(testresult.clearstatus)
{
case(IMAGE_CLEAR):
	fprintf(fp,"Image is clear\n");
	nLabelNorm++;
	break;
case(IMAGE_NOTCLEAR):
	fprintf(fp,"Image is not clear\n");
	nNotClearNum++;
	break;
case(IMAGE_HEVNOTCLEAR):
	fprintf(fp,"Image is heavy not clear \n");
	nHeavyNotClearNum++;
	break;
default:
	break;
}

switch(testresult.Noisestatus)
{
case(IMAGE_UNNOISE):
	fprintf(fp,"Image is not noise\n");
	nLabelNorm++;
	break;
case(IMAGE_NOISE):
	fprintf(fp,"Image is    noised\n");
	nNoiseNum++;
	break;
case(IMAGE_HEAVENOISE):
	fprintf(fp,"Image is heavy noise\n");
```

```
        nHeavyNoiseNum++;
        break;
default:
        break;
}

switch(testresult.lightstatus)
{
case(IMAGE_TOODARK):
        fprintf(fp,"Image is too dark\n");
        nTooDrakNum++;
        break;
case(IMAGE_TOOLIGHT):
        fprintf(fp,"Image is too light\n");
        nTooLightNum++;
        break;
default:
        nLabelNorm++;
        break;
}

switch(testresult.caststatus)
{
case(IMAGE_CAST):
        fprintf(fp,"Image is cast\n");
        nCastNum++;
        break;
default:
        nLabelNorm++;
        break;
}

switch(testresult.signalLoststatus)
{
case (IMAGE_SIGNALLOST):
        fprintf(fp,"Signal is lost\n");
        nLostNum++;
        break;
default:
        nLabelNorm++;
        break;
}
switch(testresult.imgFrozenstatus)
{
```

```
            case(IMAGE_FROZEN):
                fprintf(fp,"Image is frozen\n");
                nFrozemNum++;
                break;
            default:
                nLabelNorm++;
                break;
            }

            if (nLabelNorm==6)
            {
                nNormNum++;
            }
            fprintf(fp,"noise level = %10f\n",p1.NOISE.NOISELEVEL);
            fprintf(fp,"clear level = %10f\n",p1.CLEAR.CLEARLEVEL);
            fprintf(fp,"luminance saturation = %10f\n",p1.Bright.BrightLevel1);
            //fprintf(fp,"coverDust level = %10f\n",p1.CoverDustLevel);
            fprintf(fp,"\n\n");
        }

        fprintf(fp,"NotClearNum= %d\n",nNotClearNum);
        fprintf(fp,"HeavyNotClearNum= %d\n",nHeavyNotClearNum);
        fprintf(fp,"NoiseNum= %d\n",nNoiseNum);
        fprintf(fp,"HeavyNoiseNum= %d\n",nHeavyNoiseNum);
        fprintf(fp,"TooLightNum= %d\n",nTooLightNum);
        fprintf(fp,"TooDrakNum= %d\n",nTooDrakNum);
        fprintf(fp,"CastNum= %d\n",nCastNum);
        fprintf(fp,"FrozemNum= %d\n",nFrozemNum);
        fprintf(fp,"LostNum= %d\n",nLostNum);
        //fprintf(fp,"CoverDustNum= %d\n",nCoverDustNum);
        fprintf(fp,"NormNum= %d\n",nNormNum);
        fprintf(fp,"Imgnum= %d\n",nImgnum);
        fclose (fp);
        HYIQ_Uninit(HYIQHandle);
        cvDestroyAllWindows();

#ifdef STATIC_MEM
        HYIQ_MemMgrDestroy(hMemMgr);
        free(pMem);
#endif

        return 0;
}
```